Homework #2

[ECE30021/ITP30002] Operating Systems

Mission

- Solve problem 1, 2, and 3
  - Solve the problems on Ubuntu Linux (VirtualBox) using vim and gcc.

- Submission
  - Submit a .tgz file containing hw1_3_update.c hw2_2.c and hw2_3.c on HISNet.

    "tar cvfz hw2_*<student_id>*_*<eng_name>*.tgz hw1_3_update.c hw2_2.c hw2_3.c"

    (Do not copy&past but type the above command)
  - After compression, please check the .tgz file by decompressing in an empty directory.

    "tar xvfz hw2_*<student_id>*_*<eng_name>*.tgz"

- Due date: PM 11:00:00, Mar. 26th

Honor Code Guidelines

- “과제”
  - 과제는 교과과정의 내용을 소화하여 실질적인 활용 능력을 갖추기 위한 교육활동이다. 학생은 모든 과제를 정직하고 성실하게 수행함으로써 과제에 의도된 지식과 기술을 얻기 위해 최선을 다해야 한다.
  - 제출된 과제물은 성적 평가에 반영되므로 공식적으로 허용되지 않은 자료나 도움을 획득, 활용, 요구, 제공하는 것을 포함하여 평가의 공정성에 영향을 미치는 모든 형태의 부정행위는 단호히 거부해야 한다.
  - 수업 내용, 공지된 지식 및 정보, 또는 과제의 요구를 이해하기 위하여 동료의 도움을 받는 것은 부정행위에 포함되지 않는다. 그러나, 과제를 해결하기 위한 모든 과정은 반드시 스스로의 힘으로 수행해야 한다.
  - 담당교수가 명시적으로 허락한 경우를 제외하고 다른 사람이 작성하였거나 인터넷 등에서 획득한 과제물, 또는 프로그램 코드의 일부, 또는 전체를 이용하는 것은 부정행위에 해당한다.
  - 자신의 과제물을 타인에게 보여주거나 빌려주는 것은 공정한 평가를 방해하고, 해당 학생의 학업 성취를 저해하는 부정행위에 해당한다.
  - 팀 과제가 아닌 경우 두 명 이상이 함께 과제를 수행하여 이를 개별적으로 제출하는 것은 부정행위에 해당한다.
  - 스스로 많은 노력을 한 후에도 버그나 문제점을 파악하지 못하여 동료의 도움을 받는 경우도 단순한 문법적 오류에 그쳐야 한다. 과제가 요구하는 design, logic, algorithm의 작성에 있어서 담당교수, TA, tutor 이외에 다른 사람의 도움을 받는 것은 부정행위에 해당한다.
  - 서로 다른 학생이 제출한 제출물간 유사도가 통상적으로 발생할 수 있는 정도를 크게 넘어서는 경우, 또는 자신이 제출한 과제물에 대하여 구체적인 설명을 하지 못하는 경우에는 부정행위로 의심받거나 판정될 수 있다.

Problem 1

- Mission: peer review of HW#1 as a group
  - Sign up study group
    - Section 01: https://docs.google.com/spreadsheets/d/13c8OSrn7-I100vbSHuemXOCIPgH5tM9L3tftIvLdJ34/edit?usp=sharing
    - Section 02: https://docs.google.com/spreadsheets/d/1ozD9CYSYENmA7nza2jXrN4himsghODde25nUyffQDOE/edit?usp=sharing
  - Have an online meeting to review HW#1
    - Review the solutions of other members and share comments to each other.
    - Each member update solution of HW#1 problem 3 (hw1_3_updated.c) individually applying the comments.
      - Mark the modified part by comments as the next page

Problem 1

- Example of updated solution.

```
int main()
{
	int i = 0;

	// previous code
	/*
	<old code>

	*/

	// updated code
	<new code>


	return 0;
}
```

Problem 2

- Search the Internet for the following structures and functions
  - struct dirent → directory 정보읽기에 사용

```
struct dirent{
    long            d_ino;      //I-노드 번호 → 삭제된 파일은 d_ino = 0
    off_t           d_off;      //offset
    unsigned short  d_reclen;   //파일 이름 길이
    char            d_name[NAME_MAX+1]; // 파일 이름
}
```

  - DIR *opendir(const char *name); → directory 이름을 인수로 받아서 해당 directory Stream을 연다.
    성공시, directory Stream Pointer 반환 → 정상 return DIR*포인터로 readdir()과 closedir()사용
    실패시, NULL
  - struct dirent *readdir(DIR *dirp); → opendir()로 얻어진 directory pointer를 통해 directory 정보를 하나씩 차례로 읽습니다.
    directory 내 파일 정보를 모두 읽었을 때 or ERROR 발생시 NULL return
    readdir()이 정상이면 struct dirent 구조체의 pointer를 return
  - int closedir(DIR *dirp);
    → directory stream을 닫습니다. DIR* 포인터를 통해 닫기!
    정상: 0, 실패: 1 return
  - struct stat → 파일의 정보를 structure로 저장

```
struct stat{
    mode_t  st_mode; //파일 타입과 퍼미션
    ino_t   st_ino;//i-node 번호
    dev_t   st_dev; //장치 번호
    dev_t   st_rdev; //특수 파일의 장치 번호
    nlink_t st_nlink; //링크 수
    uid_t   st_uid;//소유자의 USER ID
    gid_t   st_gid; //소유자의 GROUP ID
    off_t   st_size; //정규파일의 바이트 수
    time_t  st_atime; //마지막 접근 시각
    time_t  st_mtime: //마지막 수정 시각
    time_t  st_ctime; //마지막 상태 변경 시각
    long    st_blksize; // I/O 블록 크기
    long    st_blocks;//할당한 블록의 개수
};
```

  - int stat(const char *pathname, struct stat *statbuf);
    → 파일의 크기, 권한, 생성일시, 최종 변경일 등, 파일의 정보를 얻는 함수.
    Symbolic link인 파일을 path로 넘기면 그원본 파일의 정보를 얻음.
  - struct tm

```
struct tm {
    int tm_sec;  // seconds
    int tm_min;  // minutes
    int tm_hour;  // hours
    int tm_mday;  // day of the month
    int tm_mon;  // month
    int tm_year;  // year
    int tm_wday;  // day of the week
    int tm_yday;  // day in the year
    int tm_isdst;  // daylight saving time
};
```

  - struct tm *localtime(const time_t *timep);
    ↳ time_t type의 시간을 지역(local 정보)의 struct tm type으로 변환

Problem 2

- Write a program that lists the current directory.
  - Display user id, group id, size, modified time of all regular files

```
$ ls -al
total 44
drwxrwxr-x 2 callee callee  4096  3월 18 13:39 . current
drwxrwxr-x 4 callee callee  4096  3월 18 10:41 .. parent
-rwxrwxr-x 1 callee callee  8680  3월 18 13:04 hw2_2
-rw-rw-r-- 1 callee callee  1185  3월 18 12:07 hw2_2.c
-rwxrwxr-x 1 callee callee 12904  3월 18 13:04 hw2_3
-rw-rw-r-- 1 callee callee  1262  3월 18 12:58 hw2_3.c

$ ./hw2_2
[reg] hw2_2.c   uid=1000, gid=1000, 1185 bytes, mtime=2022:03:18 12:07:22
[dir] .
[reg] hw2_3     uid=1000, gid=1000, 12904 bytes, mtime=2022:03:18 13:04:46
[dir] ..
[reg] hw2_3.c   uid=1000, gid=1000, 1262 bytes, mtime=2022:03:18 12:58:43
[reg] hw2_2     uid=1000, gid=1000, 8680 bytes, mtime=2022:03:18 13:04:42
```

Problem 2

- Algorithm
  - Open directory using opendir()
  - Repeat for all directory entries
    - Read a directory entry using readdir()
    - If current entry is a directory, display directory name with prefix [dir].
    - If current entry is a regular file, display the name with prefix [reg] and display its user id, group id, size, and modified time.
      - Use stat() and localtime()
  - Close directory using closedir()

Problem 3

- Write a function FindLargestFile() that finds the largest file under the specified directory.
  - void FindLargestFile(char *start_dir, char *largest, ~~int~~ long *size);
    - start_dir: the directory to start search
    - largest: a character array to pass the pathname of the largest file.
    - size: a pointer to pass the size of the largest file.
      - When you call FindLargestFile() from outside, **size* should be zero.
  - Use the opendir(), readdir(), and closedir() system calls.

Problem 3

- Combined with the main() on the next page, the result should be as follows:

```
$ ./hw2_3 .          # search the current directory
largest file = ./hw2_3 (12904 bytes)
$ ./hw2_3 ..         # search the parent directory
largest file = ../HW#1/20220312 Homework #1.pdf (243101
bytes)
```

  - The output can vary with the directory structure and contents.

Problem 3

- main()

```
#define MAX_PATH 256
#define MAX_RESULT 100

void FindLargestFile(char *start_dir, char *largest, long *size);

int main(int argc, char *argv[])
{
        if(argc < 2){
                printf("Usage: %s <start_dir>₩n", argv[0]);
                return 0;
        }

        char *start_dir = argv[1];
        char largest[MAX_PATH] = "";
        long size = 0L;      // the initial value of size should be zero
        FindLargestFile(start_dir, largest, &size);
                                ↳ argv[1] .. or .

        if(size > 0)
                printf("largest file = %s, size = %ld₩n", largest, size);
        else
                printf("no file with size > 0.₩n");

        return 0;
}
```

Problem 3

- void FindLargestFile(char *start_dir, char *largest, int *size);
- Algorithm
  - Open directory using opendir()
  - Repeat for all directory entries
    - Read a directory entry using readdir()
    - If the current entry is a subdirectory (not “.” and “..”), search the subdirectory by recursion.

      Hint) Concatenate start_dir, “/”, and the subdirectory name to build the pathname of the subdirectory.
    - If the current entry is a regular file and its size is greater than **size*
      - Set *largest* by the pathname of the current file.
      - Update **size* by the size of the current file.
  - Close directory using closedir()